# Trabajo Práctico 0: Infraestructura básica

Hernán González, *Padrón Nro. 79.460*
gonzalezhg@yahoo.com.ar

Pablo Magnaghi, *Padrón Nro. 88.126*
pablomagnaghi@gmail.com

Enzo Guagnini, *Padrón Nro. 88.325*
enzog_m@hotmail.com

1er. Cuatrimestre de 2011
66.20 Organización de Computadoras – Práctica Martes
Facultad de Ingeniería, Universidad de Buenos Aires

19/04/2011

# 1. Introducción

Este artículo se refiere a la implementación de un programa desarrollado en lenguaje C, el cual es una versión simplificada del comando `tr` de UNIX.

El mismo copiá caracteres desde *stdin* a *stdout* realizando sustituciones o eliminando caracteres seleccionados a través de los argumentos del programa, detectando e imprimiendo errores por *stderr* en caso de haberlos.

# 2. Enunciado

## 2.1. Objetivos

Familiarizarse con las herramientas de software que usaremos en los siguientes trabajos, implementado un programa (y su correspondiente documentación) que resuelva el problema piloto que se presentará más abajo.

## 2.2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

## 2.3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes. Además, es necesario que el trabajo práctico incluya (entre otras cosas), la presentación de los resultados obtenidos, explicando, cuando corresponda, con fundamentos reales, las causas o razones de cada resultado obtenido.

El informe deberá respetar el modelo de referencia que se encuentra en el grupo, y se valorarán aquellos escritos usando la herramienta TEX / LATEX.

## 2.4. Recursos

Usaremos el programa GXemul para simular el entorno de desarrollo que utilizaremos en este y otros trabajos prácticos, una máquina MIPS corriendo una versión reciente del sistema operativo NetBSD.

## 2.5. Implementación

### 2.5.1. Programa

El programa a escribir en lenguaje C, es una versión minimalista del comando `tail` de UNIX. El mismo copia caracteres desde *stdin* a *stdout* realizando sustituciones o eliminando caracteres seleccionados a través de los argumentos del programa. La funcionalidad a desarrollar consiste en:

- Traducción de caracteres.
- Sumarización de caracteres.
- Borrado de caracteres.

El programa recibe dos argumentos, *string1* y *string2*, y los usa para operar sobre los

El programa recibe dos argumentos, *string1* y *string2*, y los usa para operar sobre los caracteres leidos de *stdin*.La traducción de caracteres se realiza a partir de la posición de un caracter leido de *stdin* en *string1*, reemplazandolo por el caracter en la misma posición de *string2*. La sumarización de caracteres se realiza a partir de los caracteres contenidos en *string1*. Si un caracter leido de *stdin* está contenido en *string1*, entonces el mismo debe escribirse en *stdout* una sola vez. Para el borrado de caracteres no se escriben en *stdout* aquellos que son leidos de *stdin* y están contenidos en *string1*. El programa devuelve 0 en caso de exito. Todos los errores deben escribirse en *stderr*.

### 2.5.2. Consideraciones

### 2.5.3. Combinación de tareas

Estas tareas pueden o no estar superpuestas de la siguiente manera:

`tp0 -s string1 string2`: La sumarización de caracteres ocurre luego de la traducción.

`tp0 -sd string1 string2`: Se eliminan los caracteres contenidos en *string1* y se sumarizan los contenidos en *string2*.

### 2.5.4. Sobre la longitud de los argumentos

En caso de que la longitud de *string1* sea mayor a la longitud de *string2*, entonces se debe extender *string2* con su último caracter hasta igualar las longitudes.

### 2.5.5. Ocurrencias en el primer argumento

En caso de que un caracter se repita en *string1*, debe tomarse la última ocurrencia al realizar la traducción.

### 2.5.6. Ejemplos

Primero, usamos la opción `-h` para ver el mensaje de ayuda:

```
$ tp0 -h
Usage:
tp0 -h
tp0 -V
tp0 [options] string1 string2
tp0 [options] string1
Options:
-V, --version Print version and quit.
-h, --help Print this information and quit.
-d, --delete Delete characters in string1
-s, --squeeze Squeeze characters in input.

Examples:

tp0 abc 123 <alpha.txt
tp0 -ds 123456789 aeiou

$ echo ‘‘3.14159192’’ | tp0 1 9
3.94959992

$ tp0 aba 123 <text.txt
323

$ echo ‘‘aa bb cc’’ | tp0 -s ab
a b cc

$ echo ‘‘aa bb cc’’ | tp0 -s ab 12
1 2 cc

$ echo ‘‘Hola mundo’’ |tp0 -d Ho
la mund

$ echo ‘‘aabbcc’’ | tp0 -ds a b
bcc

$ echo ‘‘aaaa’’ | tp0 -s a b
b
```

#### 2.5.7. Consideraciones

Ademas de la funcionalidad descripta en 5.1, puede implementarse los siguientes items:

- Opción `-c`: COmplemento de los caracteres en el *string1*.
- Conjunto de caracteres.

#### 2.5.8. Portabilidad

Como es usual, es necesario que la implementación desarrollada provea un grado mínimo de portabilidad. Para satisfacer esto, el programa deberá funcionar al menos en NetBSD/pmax (usando el simulador GXEmul) y la versión de Linux usada para correr el simulador.

### 2.6. Informe

El informe deberá incluir:

- Documentación relevante al diseño e implementación del programa.
- Comando(s) para compilar el programa.
- Corridas de prueba y sus correspondientes comentarios.
- El código fuente completo.
- El código MIPS32 generado por el compilador.
- Este enunciado.

## 3. Documentación y diseño

La realización de este informe se hizo con la herramienta TEX / LATEX. Tanto el arhivo en formato pdf como el archivo .tex se encuentran en la carpeta doc del cd entregado.

Para el diseño se utilizó un parseador, creado por nosotros, que sirve para analizar los argumentos que se le pasan al programa a través de la línea de comandos.

Luego se realiza la función indicada y el programa devuelve errores en el caso correspondiente a través de *stderr*.

## 4. Códigos de retorno y mensajes de error

Si la ejecución fue exitosa el código de retorno es 0 (cero), de lo contrario se informa por *stderr* el error ocurrido y se retorna un -1 (menos uno).

# 5. Casos de prueba

Los casos de prueba utilizados son los que figuran a continuación.

```
$ echo ‘‘3.14159192’’ | ./tp0 1 9
3.94959992

$ echo ‘‘aba’’ | ./tp0 aba 123
323

$ echo ‘‘aa bb cc’’ | ./tp0 -s ab
a b cc

$ echo ‘‘aa bb cc’’ | ./tp0 -s ab 12
1 2 cc

$ echo ‘‘Hola mundo’’ | ./tp0 -d Ho
la mund

$ echo ‘‘aabbcc’’ | ./tp0 -ds a b
bcc

$ echo ‘‘aaaa’’ | ./tp0 -s a b
b

$ echo ‘‘bbb bbb’’ | ./tp0 -s b
b b

$echo ‘‘bbbaaabb bbabbbbbbb’’ | ./tp0 -ds b a
a a

$echo ‘‘f’’ | ./tp0 aba def
f

$echo ‘‘abaco’’ | ./tp0 -s a
abaco

$echo ‘‘abbbbbabbaaaco’’ | ./tp0 -s a b
bco

$echo ‘‘bbbbbbaaaaaaabbb’’ | ./tp0 -ds a b
b
$echo ‘‘abaco baco abaco’’ | ./tp0 a b
bbbco bbco bbbco
```

# 6. Instalación

La instalación se encuentra automatizada por medio de un script de shell. Para instalar el software se debe proceder de la siguiente manera:

- Descomprimir el archivo en la carpeta donde se desee realizar la instalación.
- En el shell cambiar el directorio de trabajo actual hacia el directorio de instalación.
- Ingresar la orden $sh ./run.sh. Esto dispara el script que realiza la compilación e instalación del software. El binario es instalado en la carpeta bin del cd entregado.

# 7. Código fuente

A contiuación se detalla el código fuente implementado en lenguaje C que se encuentra en carpeta code del cd entregado:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <strings.h>
5
6  #define UNO 1
7  #define DOS 2
8  #define TRES 3
9  #define CUATRO 4
10 #define EXITO 0
11 #define ERROR -1
12 #define NCHARS 256
13 #define ES_D 0
14 #define ES_S 1
15 #define ES_DS 2
16
17 typedef enum boolean { FALSE, TRUE } boolean;
18
19 #define ASCII0 0x00
20
21 int dString1(char*);
22 int sString1(char*);
23 int sString1String2(char*, char*);
24 int dsString1String2(char*, char*);
25 int string1String2(char *, char *);
26 int opcionArgcDos(char*);
27 int opcionArgcTres(char*, char*);
28 int opcionArgcCuatro(char*, char*, char*);
29 void obtenerOpcion(char*, int*, boolean*);
30 void inicializarArrayBoolean(boolean*);
31 void mostrarH();
32 int errorPorEscritura();
33
34 int main(int argc, char **argv) {
35     int retorno = 0;
36     switch (argc) {
37         case DOS:    retorno = opcionArgcDos(argv[1]);
38                      break;
39         case TRES:   retorno = opcionArgcTres(argv[1], argv[2]);
40                      break;
41         case CUATRO: retorno = opcionArgcCuatro(argv[1], argv[2], argv
                           [3]);
42                      break;
43         default:     mostrarH();
44                      break;
45     }
46     return retorno;
47 }
48
49 // LLama a la funcion correspondiente dependiendo de los argumentos
50 // Opciones validas:
51 // ./tp0 -V
52 // ./tp0 -h
53
54 int opcionArgcDos(char *option) {
55     char opcion1[] = "-V";
56     char opcion2[] = "-h";
57
58     if ( strcmp(opcion1, option) == 0 ) {
59         printf ("tp0 Version 19/04/2011\n");
60         return EXITO;
61     }
62
63     if ( strcmp(opcion2, option) == 0 ) {
64         mostrarH();
65         return EXITO;
66     }
```

```
67
68      perror("ERROR: Los argumentos validos son -V o -h\n");
69      return ERROR;
70  }
71
72  // LLama a la funcion correspondiente dependiendo de los argumentos
73  // Opciones validas:
74  // ./tp0 -d string1
75  // ./tp0 -s string1
76  // ./tp0 string1 string2
77
78  int opcionArgcTres(char *string1, char *string2) {
79      boolean esOpcion = FALSE;
80      int numFuncion = 0;
81
82      int (*pFuncion[DOS])(char*) = {dString1, sString1};
83
84      obtenerOpcion(string1, &numFuncion, &esOpcion);
85
86      if (esOpcion) {
87          if (numFuncion != ES_DS){
88              return (*pFuncion[numFuncion])(string2);
89          }
90
91          perror("ERROR: La opcion -ds debe recibir 2 cadenas");
92          return ERROR;
93
94      }
95      // si la primera cadena no es un opcion -s, -d, -ds.
96      // Se toma como otra cadena y por defecto la opcion es sustituir
            string1String2
97      return string1String2(string1, string2);
98  }
99
100 // LLama a la funcion Correspondiente dependiendo de los argumentos
101 // Opciones validas:
102 // ./tp0 -s string1 string2
103 // ./tp0 -ds string1 string2
104
105 int opcionArgcCuatro(char *option, char *string1, char *string2) {
106     boolean esOpcion = FALSE;
107     int numFuncion = 0;
108
109     int (*pFuncion[DOS])(char*, char*) = {sString1String2,
            dsString1String2 };
110
111     obtenerOpcion(option, &numFuncion, &esOpcion);
112
113     if ( esOpcion ) {
114         if (numFuncion != ES_D){
115             return (*pFuncion[numFuncion-1])(string1, string2);
116         }
117         perror("ERROR: La opcion -d debe recibir 1 cadena");
118         return ERROR;
119     }
120     perror("ERROR: Sintaxis invalida");
121     return ERROR;
122 }
123
124 // Verifica cual de las opciones se ingreso (-d,-s,-ds);
125
126 void obtenerOpcion(char *option, int *numFuncion, boolean *esOpcion) {
127     char opcion1[] = "-d";
128     char opcion2[] = "-s";
129     char opcion3[] = "-ds";
130
131     if (strcmp(opcion1, option) == 0 ) {
132         *esOpcion = TRUE;
133         *numFuncion = ES_D;
134     } else if (strcmp(opcion2, option) == 0 ) {
135         *esOpcion = TRUE;
136         *numFuncion = ES_S;
137     } else if (strcmp(opcion3, option) == 0 ) {
138         *esOpcion = TRUE;
```

```
139             *numFuncion = ES_DS;
140         }
141     }
142
143     // Cuando el usuario hace una llamada del tipo ./tr -d string1
144     // Elimina los caracteres de stdin segun string1 y muestra el resultado
            por stdout
145
146     int dString1(char *string1) {
147         int ch, i = 0;
148         int size = strlen(string1);
149         boolean debeBorrarse[NCHARS];
150         inicializarArrayBoolean(debeBorrarse);
151
152         for (i = 0; (i < size); i++){
153             debeBorrarse[((int)string1[i])] = TRUE;
154         }
155
156         while ((ch = getchar()) != EOF) {
157             if (!debeBorrarse[ch])
158                 if(putchar(ch) == EOF)
159                     return errorPorEscritura();
160         }
161         return EXITO;
162     }
163
164     // Cuando el usuario hace una llamada del tipo ./tr -s string1
165     // Se hace la sumarizacion, se usa un array de boolean para saber si
166     // el caracter ya fue leido.
167     int sString1(char *string1) {
168         int i = 0, ch = 0, lastCh = -1;
169         boolean sumarizar[NCHARS];
170         inicializarArrayBoolean(sumarizar);
171
172         int size = strlen(string1);
173
174         for (i=0;i<size;i++){
175             sumarizar[(int)string1[i]] = TRUE;
176         }
177         if (lastCh == -1){
178             if((ch = getchar()) == EOF)
179                 return ERROR;
180             if(putchar(ch) == EOF)
181                 return errorPorEscritura();
182             lastCh = ch;
183         }
184         while ((ch = getchar()) != EOF) {
185             if ((!sumarizar[ch]) || (lastCh != ch)){
186                 if(putchar(ch) == EOF)
187                     return errorPorEscritura();
188                 lastCh = ch;
189             }
190         }
191         return EXITO;
192     }
193
194     // Cuando el usuario hace una llamada del tipo ./tr string1 string2
195     // Sustituye los caracteres de string1 por los de string2
196     // Si la longitud de string2 es menor a la de string1 se reemplaza por
            el
197     // ultimo caracter de string2 en el caso en que ya se supere el tamanio
198     int string1String2(char *string1, char *string2){
199         int ch, i, final;
200         char cambios[NCHARS];
201         bzero(cambios,NCHARS);
202
203         int string1Size = strlen(string1);
204         int string2Size = strlen(string2);
205
206         if (string1Size <= string2Size) final = string1Size;
207             else final = string2Size;
208
209         for (i=0; i < final; i++){
210             cambios[((int)string1[i])] = string2[i];
```

```
211         }
212
213         if (final == string2Size){ //extiendo con el ultimo caracter de
                string 2
214             for (i = string2Size; i < string1Size; i++)
215                 cambios[((int)string1[i])] = string2[string2Size-1];
216         }
217
218         while ((ch = getchar()) != EOF) {
219             if (cambios[ch] == ASCII0) {
220                 if (putchar(ch) == EOF)
221                     return errorPorEscritura();
222             } else {
223                 if (putchar(cambios[ch]) == EOF)
224                     return errorPorEscritura();
225             }
226         }
227         return EXITO;
228     }
229
230     // Cuando el usuario hace una llamada del tipo ./tr -s string1 string2.
231     //remplaza los caracteres de string 1 por los de 2, extendiendo si es
            nesesario
232     //y sumariza segun string 2
233     int sString1String2(char *string1, char *string2) {
234         int ch = 0, i = 0, lastCh = -1, final = 0;
235         char cambios[NCHARS];
236         bzero(cambios,NCHARS);
237
238         int string1Size = strlen(string1);
239         int string2Size = strlen(string2);
240
241         if (string1Size <= string2Size) final = string1Size;
242             else final = string2Size;
243
244         for (i=0; i < final; i++){
245             cambios[((int)string1[i])] = string2[i];
246         }
247
248         if (final == string2Size){ //extiendo con el ultimo caracter de
                string 2
249             for (i = string2Size; i < string1Size; i++)
250                 cambios[((int)string1[i])] = string2[string2Size-1];
251         }
252
253         while ((ch = getchar()) != EOF) {
254             if (cambios[ch] == ASCII0){
255                 if (cambios[lastCh] != ch) {
256                     if (putchar(ch) == EOF)
257                         return errorPorEscritura();
258                     lastCh = ch;
259                 }
260             } else {
261                 if (cambios[ch] != cambios[lastCh]){
262                     if (putchar(cambios[ch]) == EOF)
263                         return errorPorEscritura();
264                     lastCh = ch;
265                 }
266             }
267         }
268         return EXITO;
269     }
270
271     // Cuando el usuario hace una llamada del tipo ./tr -ds cadena1 cadena2
272     int dsString1String2(char *string1, char *string2) {
273         int ch, i, lastCh = 0;
274         boolean debeBorrarse[NCHARS],sumarizar[NCHARS];
275
276         inicializarArrayBoolean(debeBorrarse);
277         inicializarArrayBoolean(sumarizar);
278
279         int string1Size = strlen(string1);
280         int string2Size = strlen(string2);
281
```

```
282     for (i = 0; i < string1Size; i++){
283         debeBorrarse[((int)string1[i])] = TRUE;
284     }
285
286     for (i=0;i<string2Size;i++){
287             sumarizar[(int)string2[i]] = TRUE;
288     }
289
290     while ((ch = getchar()) != EOF) {
291         if (!debeBorrarse[ch]) {
292             if ((!sumarizar[ch]) || (lastCh != ch)){
293                 if(putchar(ch) == EOF)
294                     return errorPorEscritura();
295                 lastCh = ch;
296             }
297         }
298     }
299     return EXITO;
300 }
301
302 void inicializarArrayBoolean(boolean *fueLeido){
303     int i;
304     for(i = 0; i < 256; i++){
305         fueLeido[i] = FALSE;
306     }
307 }
308
309 void mostrarH() {
310     printf ("Usage:\n");
311     printf ("tp0 -h\n");
312     printf ("tp0 -V\n");
313     printf ("tp0 [options] string1 string2\n");
314     printf ("tp0 [options] string1\n");
315     printf ("Options:\n");
316     printf ("-V, --version\n");
317     printf ("-h, --help\n");
318     printf ("-d, --delete\n");
319     printf ("-s, --squeeze\n");
320 }
321
322 int errorPorEscritura() {
323     perror("ERROR: ocurre un error de escritura");
324     return ERROR;
325 }
```

## 7.1. Código MIPS generado por el compilador GCC

El siguiente es el código assembly generado en el emulador *GXemul*, corriendo una máquina de arquitectura *MIPS* con el sistema operativo *NetBSD* que se encuentra en la carpeta code del cd entregado:

```
1       .file   1 "tp0.c"
2       .section .mdebug.abi32
3       .previous
4       .abicalls
5       .text
6       .align  2
7       .globl  main
8       .ent    main
9   main:
10      .frame  $fp,48,$ra      # vars= 8, regs= 3/0, args= 16, extra= 8
11      .mask   0xd0000000,-8
12      .fmask  0x00000000,0
13      .set    noreorder
14      .cpload $t9
15      .set    reorder
16      subu    $sp,$sp,48
17      .cprestore 16
18      sw  $ra,40($sp)
19      sw  $fp,36($sp)
20      sw  $gp,32($sp)
```

```
21      move    $fp,$sp
22      sw  $a0,48($fp)
23      sw  $a1,52($fp)
24      sw  $zero,24($fp)
25      lw  $v0,48($fp)
26      sw  $v0,28($fp)
27      li  $v0,3               # 0x3
28      lw  $v1,28($fp)
29      beq $v1,$v0,$L20
30      lw  $v1,28($fp)
31      slt $v0,$v1,4
32      beq $v0,$zero,$L24
33      li  $v0,2               # 0x2
34      lw  $v1,28($fp)
35      beq $v1,$v0,$L19
36      b   $L22
37  $L24:
38      li  $v0,4               # 0x4
39      lw  $v1,28($fp)
40      beq $v1,$v0,$L21
41      b   $L22
42  $L19:
43      lw  $v0,52($fp)
44      addu    $v0,$v0,4
45      lw  $a0,0($v0)
46      la  $t9,opcionArgcDos
47      jal $ra,$t9
48      sw  $v0,24($fp)
49      b   $L18
50  $L20:
51      lw  $v0,52($fp)
52      addu    $v1,$v0,4
53      lw  $v0,52($fp)
54      addu    $v0,$v0,8
55      lw  $a0,0($v1)
56      lw  $a1,0($v0)
57      la  $t9,opcionArgcTres
58      jal $ra,$t9
59      sw  $v0,24($fp)
60      b   $L18
61  $L21:
62      lw  $v0,52($fp)
63      addu    $a0,$v0,4
64      lw  $v0,52($fp)
65      addu    $v1,$v0,8
66      lw  $v0,52($fp)
67      addu    $v0,$v0,12
68      lw  $a0,0($a0)
69      lw  $a1,0($v1)
70      lw  $a2,0($v0)
71      la  $t9,opcionArgcCuatro
72      jal $ra,$t9
73      sw  $v0,24($fp)
74      b   $L18
75  $L22:
76      la  $t9,mostrarH
77      jal $ra,$t9
78  $L18:
79      lw  $v0,24($fp)
80      move    $sp,$fp
81      lw  $ra,40($sp)
82      lw  $fp,36($sp)
83      addu    $sp,$sp,48
84      j   $ra
85      .end    main
86      .size   main, .-main
87      .rdata
88      .align  2
89  $LC0:
90      .ascii  "-V\000"
91      .align  2
92  $LC1:
93      .ascii  "-h\000"
94      .align  2
```

```
95  $LC2:
96      .ascii  "tp0 Version 19/04/2011\n\000"
97      .align  2
98  $LC3:
99      .ascii  "ERROR: Los argumentos validos son -V o -h\n\000"
100     .text
101     .align  2
102     .globl  opcionArgcDos
103     .ent    opcionArgcDos
104 opcionArgcDos:
105     .frame  $fp,64,$ra      # vars= 24, regs= 3/0, args= 16, extra= 8
106     .mask   0xd0000000,-8
107     .fmask  0x00000000,0
108     .set    noreorder
109     .cpload $t9
110     .set    reorder
111     subu    $sp,$sp,64
112     .cprestore 16
113     sw  $ra,56($sp)
114     sw  $fp,52($sp)
115     sw  $gp,48($sp)
116     move    $fp,$sp
117     sw  $a0,64($fp)
118     lhu $v0,$LC0
119     sh  $v0,24($fp)
120     lbu $v0,$LC0+2
121     sb  $v0,26($fp)
122     lhu $v0,$LC1
123     sh  $v0,32($fp)
124     lbu $v0,$LC1+2
125     sb  $v0,34($fp)
126     addu    $a0,$fp,24
127     lw  $a1,64($fp)
128     la  $t9,strcmp
129     jal $ra,$t9
130     bne $v0,$zero,$L26
131     la  $a0,$LC2
132     la  $t9,printf
133     jal $ra,$t9
134     sw  $zero,40($fp)
135     b   $L25
136 $L26:
137     addu    $v0,$fp,32
138     move    $a0,$v0
139     lw  $a1,64($fp)
140     la  $t9,strcmp
141     jal $ra,$t9
142     bne $v0,$zero,$L27
143     la  $t9,mostrarH
144     jal $ra,$t9
145     sw  $zero,40($fp)
146     b   $L25
147 $L27:
148     la  $a0,$LC3
149     la  $t9,perror
150     jal $ra,$t9
151     li  $v0,-1          # 0xffffffffffffffff
152     sw  $v0,40($fp)
153 $L25:
154     lw  $v0,40($fp)
155     move    $sp,$fp
156     lw  $ra,56($sp)
157     lw  $fp,52($sp)
158     addu    $sp,$sp,64
159     j   $ra
160     .end    opcionArgcDos
161     .size   opcionArgcDos, .-opcionArgcDos
162     .data
163     .align  2
164 $LC4:
165     .word   dString1
166     .word   sString1
167     .rdata
168     .align  2
```

```
169 $LC5:
170     .ascii  "ERROR: La opcion -ds debe recibir 2 cadenas\000"
171     .text
172     .align  2
173     .globl  opcionArgcTres
174     .ent    opcionArgcTres
175 opcionArgcTres:
176     .frame  $fp,64,$ra      # vars= 24, regs= 3/0, args= 16, extra= 8
177     .mask   0xd0000000,-8
178     .fmask  0x00000000,0
179     .set    noreorder
180     .cpload $t9
181     .set    reorder
182     subu    $sp,$sp,64
183     .cprestore 16
184     sw  $ra,56($sp)
185     sw  $fp,52($sp)
186     sw  $gp,48($sp)
187     move    $fp,$sp
188     sw  $a0,64($fp)
189     sw  $a1,68($fp)
190     sw  $zero,24($fp)
191     sw  $zero,28($fp)
192     lw  $v0,$LC4
193     sw  $v0,32($fp)
194     lw  $v0,$LC4+4
195     sw  $v0,36($fp)
196     addu    $v0,$fp,28
197     lw  $a0,64($fp)
198     move    $a1,$v0
199     addu    $a2,$fp,24
200     la  $t9,obtenerOpcion
201     jal $ra,$t9
202     lw  $v0,24($fp)
203     beq $v0,$zero,$L29
204     lw  $v1,28($fp)
205     li  $v0,2               # 0x2
206     beq $v1,$v0,$L30
207     lw  $v0,28($fp)
208     sll $v1,$v0,2
209     addu    $v0,$fp,24
210     addu    $v0,$v1,$v0
211     addu    $v0,$v0,8
212     lw  $v0,0($v0)
213     lw  $a0,68($fp)
214     move    $t9,$v0
215     jal $ra,$t9
216     sw  $v0,40($fp)
217     b   $L28
218 $L30:
219     la  $a0,$LC5
220     la  $t9,perror
221     jal $ra,$t9
222     li  $v0,-1              # 0xffffffffffffffff
223     sw  $v0,40($fp)
224     b   $L28
225 $L29:
226     lw  $a0,64($fp)
227     lw  $a1,68($fp)
228     la  $t9,string1String2
229     jal $ra,$t9
230     sw  $v0,40($fp)
231 $L28:
232     lw  $v0,40($fp)
233     move    $sp,$fp
234     lw  $ra,56($sp)
235     lw  $fp,52($sp)
236     addu    $sp,$sp,64
237     j   $ra
238     .end    opcionArgcTres
239     .size   opcionArgcTres, .-opcionArgcTres
240     .data
241     .align  2
242 $LC6:
```

```
243       .word   sString1String2
244       .word   dsString1String2
245       .rdata
246       .align  2
247   $LC7:
248       .ascii  "ERROR: La opcion -d debe recibir 1 cadena\000"
249       .align  2
250   $LC8:
251       .ascii  "ERROR: Sintaxis invalida\000"
252       .text
253       .align  2
254       .globl  opcionArgcCuatro
255       .ent    opcionArgcCuatro
256   opcionArgcCuatro:
257       .frame  $fp,64,$ra      # vars= 24, regs= 3/0, args= 16, extra= 8
258       .mask   0xd0000000,-8
259       .fmask  0x00000000,0
260       .set    noreorder
261       .cpload $t9
262       .set    reorder
263       subu    $sp,$sp,64
264       .cprestore 16
265       sw  $ra,56($sp)
266       sw  $fp,52($sp)
267       sw  $gp,48($sp)
268       move    $fp,$sp
269       sw  $a0,64($fp)
270       sw  $a1,68($fp)
271       sw  $a2,72($fp)
272       sw  $zero,24($fp)
273       sw  $zero,28($fp)
274       lw  $v0,$LC6
275       sw  $v0,32($fp)
276       lw  $v0,$LC6+4
277       sw  $v0,36($fp)
278       addu    $v0,$fp,28
279       lw  $a0,64($fp)
280       move    $a1,$v0
281       addu    $a2,$fp,24
282       la  $t9,obtenerOpcion
283       jal $ra,$t9
284       lw  $v0,24($fp)
285       beq $v0,$zero,$L32
286       lw  $v0,28($fp)
287       beq $v0,$zero,$L33
288       lw  $v0,28($fp)
289       sll $v1,$v0,2
290       addu    $v0,$fp,24
291       addu    $v0,$v1,$v0
292       addu    $v0,$v0,4
293       lw  $v0,0($v0)
294       lw  $a0,68($fp)
295       lw  $a1,72($fp)
296       move    $t9,$v0
297       jal $ra,$t9
298       sw  $v0,40($fp)
299       b   $L31
300   $L33:
301       la  $a0,$LC7
302       la  $t9,perror
303       jal $ra,$t9
304       li  $v0,-1          # 0xffffffffffffffff
305       sw  $v0,40($fp)
306       b   $L31
307   $L32:
308       la  $a0,$LC8
309       la  $t9,perror
310       jal $ra,$t9
311       li  $v0,-1          # 0xffffffffffffffff
312       sw  $v0,40($fp)
313   $L31:
314       lw  $v0,40($fp)
315       move    $sp,$fp
316       lw  $ra,56($sp)
```

```
317       lw   $fp,52($sp)
318       addu     $sp,$sp,64
319       j    $ra
320       .end     opcionArgcCuatro
321       .size    opcionArgcCuatro, .-opcionArgcCuatro
322       .rdata
323       .align   2
324   $LC9:
325       .ascii   "-d\000"
326       .align   2
327   $LC10:
328       .ascii   "-s\000"
329       .align   2
330   $LC11:
331       .ascii   "-ds\000"
332       .text
333       .align   2
334       .globl   obtenerOpcion
335       .ent     obtenerOpcion
336   obtenerOpcion:
337       .frame   $fp,64,$ra      # vars= 24, regs= 3/0, args= 16, extra= 8
338       .mask    0xd0000000,-8
339       .fmask   0x00000000,0
340       .set     noreorder
341       .cpload  $t9
342       .set     reorder
343       subu     $sp,$sp,64
344       .cprestore 16
345       sw   $ra,56($sp)
346       sw   $fp,52($sp)
347       sw   $gp,48($sp)
348       move     $fp,$sp
349       sw   $a0,64($fp)
350       sw   $a1,68($fp)
351       sw   $a2,72($fp)
352       lhu  $v0,$LC9
353       sh   $v0,24($fp)
354       lbu  $v0,$LC9+2
355       sb   $v0,26($fp)
356       lhu  $v0,$LC10
357       sh   $v0,32($fp)
358       lbu  $v0,$LC10+2
359       sb   $v0,34($fp)
360       lw   $v0,$LC11
361       sw   $v0,40($fp)
362       addu     $a0,$fp,24
363       lw   $a1,64($fp)
364       la   $t9,strcmp
365       jal  $ra,$t9
366       bne  $v0,$zero,$L35
367       lw   $v0,72($fp)
368       li   $v1,1               # 0x1
369       sw   $v1,0($v0)
370       lw   $v0,68($fp)
371       sw   $zero,0($v0)
372       b    $L34
373   $L35:
374       addu     $v0,$fp,32
375       move     $a0,$v0
376       lw   $a1,64($fp)
377       la   $t9,strcmp
378       jal  $ra,$t9
379       bne  $v0,$zero,$L37
380       lw   $v1,72($fp)
381       li   $v0,1               # 0x1
382       sw   $v0,0($v1)
383       lw   $v1,68($fp)
384       li   $v0,1               # 0x1
385       sw   $v0,0($v1)
386       b    $L34
387   $L37:
388       addu     $v0,$fp,40
389       move     $a0,$v0
390       lw   $a1,64($fp)
```

```
391       la   $t9,strcmp
392       jal  $ra,$t9
393       bne  $v0,$zero,$L34
394       lw   $v1,72($fp)
395       li   $v0,1            # 0x1
396       sw   $v0,0($v1)
397       lw   $v1,68($fp)
398       li   $v0,2            # 0x2
399       sw   $v0,0($v1)
400   $L34:
401       move     $sp,$fp
402       lw   $ra,56($sp)
403       lw   $fp,52($sp)
404       addu     $sp,$sp,64
405       j    $ra
406       .end     obtenerOpcion
407       .size    obtenerOpcion, .-obtenerOpcion
408       .align   2
409       .globl   dString1
410       .ent     dString1
411   dString1:
412       .frame   $fp,1088,$ra         # vars= 1048, regs= 3/0, args= 16, extra
              = 8
413       .mask    0xd0000000,-8
414       .fmask   0x00000000,0
415       .set     noreorder
416       .cpload  $t9
417       .set     reorder
418       subu     $sp,$sp,1088
419       .cprestore 16
420       sw   $ra,1080($sp)
421       sw   $fp,1076($sp)
422       sw   $gp,1072($sp)
423       move     $fp,$sp
424       sw   $a0,1088($fp)
425       sw   $zero,28($fp)
426       lw   $a0,1088($fp)
427       la   $t9,strlen
428       jal  $ra,$t9
429       sw   $v0,32($fp)
430       addu     $v0,$fp,40
431       move     $a0,$v0
432       la   $t9,inicializarArrayBoolean
433       jal  $ra,$t9
434       sw   $zero,28($fp)
435   $L41:
436       lw   $v0,28($fp)
437       lw   $v1,32($fp)
438       slt  $v0,$v0,$v1
439       bne  $v0,$zero,$L44
440       b    $L42
441   $L44:
442       lw   $v1,1088($fp)
443       lw   $v0,28($fp)
444       addu     $v0,$v1,$v0
445       lb   $v0,0($v0)
446       sll  $v1,$v0,2
447       addu     $v0,$fp,24
448       addu     $v0,$v1,$v0
449       addu     $v1,$v0,16
450       li   $v0,1            # 0x1
451       sw   $v0,0($v1)
452       lw   $v0,28($fp)
453       addu     $v0,$v0,1
454       sw   $v0,28($fp)
455       b    $L41
456   $L42:
457       .set     noreorder
458       nop
459       .set     reorder
460   $L45:
461       lw   $v0,__sF+4
462       addu     $v0,$v0,-1
463       sw   $v0,__sF+4
```

```
464       bgez    $v0,$L48
465       la  $a0,__sF
466       la  $t9,__srget
467       jal $ra,$t9
468       sw  $v0,1068($fp)
469       b   $L49
470   $L48:
471       la  $v0,__sF
472       lw  $v1,0($v0)
473       move    $a0,$v1
474       lbu $a0,0($a0)
475       sw  $a0,1068($fp)
476       addu    $v1,$v1,1
477       sw  $v1,0($v0)
478   $L49:
479       lw  $v1,1068($fp)
480       sw  $v1,24($fp)
481       li  $v0,-1          # 0xffffffffffffffff
482       bne $v1,$v0,$L47
483       b   $L46
484   $L47:
485       lw  $v0,24($fp)
486       sll $v1,$v0,2
487       addu    $v0,$fp,24
488       addu    $v0,$v1,$v0
489       addu    $v0,$v0,16
490       lw  $v0,0($v0)
491       bne $v0,$zero,$L45
492       lw  $a0,24($fp)
493       la  $a1,__sF+88
494       la  $t9,__sputc
495       jal $ra,$t9
496       move    $v1,$v0
497       li  $v0,-1          # 0xffffffffffffffff
498       bne $v1,$v0,$L45
499       la  $t9,errorPorEscritura
500       jal $ra,$t9
501       sw  $v0,1064($fp)
502       b   $L40
503   $L46:
504       sw  $zero,1064($fp)
505   $L40:
506       lw  $v0,1064($fp)
507       move    $sp,$fp
508       lw  $ra,1080($sp)
509       lw  $fp,1076($sp)
510       addu    $sp,$sp,1088
511       j   $ra
512       .end    dString1
513       .size   dString1, .-dString1
514       .align  2
515       .globl  sString1
516       .ent    sString1
517   sString1:
518       .frame  $fp,1096,$ra        # vars= 1056, regs= 3/0, args= 16, extra
              = 8
519       .mask   0xd0000000,-8
520       .fmask  0x00000000,0
521       .set    noreorder
522       .cpload $t9
523       .set    reorder
524       subu    $sp,$sp,1096
525       .cprestore 16
526       sw  $ra,1088($sp)
527       sw  $fp,1084($sp)
528       sw  $gp,1080($sp)
529       move    $fp,$sp
530       sw  $a0,1096($fp)
531       sw  $zero,24($fp)
532       sw  $zero,28($fp)
533       li  $v0,-1          # 0xffffffffffffffff
534       sw  $v0,32($fp)
535       addu    $v0,$fp,40
536       move    $a0,$v0
```

```
537       la   $t9,inicializarArrayBoolean
538       jal  $ra,$t9
539       lw   $a0,1096($fp)
540       la   $t9,strlen
541       jal  $ra,$t9
542       sw   $v0,1064($fp)
543       sw   $zero,24($fp)
544   $L53:
545       lw   $v0,24($fp)
546       lw   $v1,1064($fp)
547       slt  $v0,$v0,$v1
548       bne  $v0,$zero,$L56
549       b    $L54
550   $L56:
551       lw   $v1,1096($fp)
552       lw   $v0,24($fp)
553       addu     $v0,$v1,$v0
554       lb   $v0,0($v0)
555       sll  $v1,$v0,2
556       addu     $v0,$fp,24
557       addu     $v0,$v1,$v0
558       addu     $v1,$v0,16
559       li   $v0,1               # 0x1
560       sw   $v0,0($v1)
561       lw   $v0,24($fp)
562       addu     $v0,$v0,1
563       sw   $v0,24($fp)
564       b    $L53
565   $L54:
566       lw   $v1,32($fp)
567       li   $v0,-1              # 0xffffffffffffffff
568       bne  $v1,$v0,$L57
569       lw   $v0,__sF+4
570       addu     $v0,$v0,-1
571       sw   $v0,__sF+4
572       bgez     $v0,$L59
573       la   $a0,__sF
574       la   $t9,__srget
575       jal  $ra,$t9
576       sw   $v0,1072($fp)
577       b    $L60
578   $L59:
579       la   $v0,__sF
580       lw   $v1,0($v0)
581       move     $a0,$v1
582       lbu  $a0,0($a0)
583       sw   $a0,1072($fp)
584       addu     $v1,$v1,1
585       sw   $v1,0($v0)
586   $L60:
587       lw   $v1,1072($fp)
588       sw   $v1,28($fp)
589       li   $v0,-1              # 0xffffffffffffffff
590       bne  $v1,$v0,$L58
591       li   $v0,-1              # 0xffffffffffffffff
592       sw   $v0,1068($fp)
593       b    $L52
594   $L58:
595       lw   $a0,28($fp)
596       la   $a1,__sF+88
597       la   $t9,__sputc
598       jal  $ra,$t9
599       move     $v1,$v0
600       li   $v0,-1              # 0xffffffffffffffff
601       bne  $v1,$v0,$L61
602       la   $t9,errorPorEscritura
603       jal  $ra,$t9
604       sw   $v0,1068($fp)
605       b    $L52
606   $L61:
607       lw   $v0,28($fp)
608       sw   $v0,32($fp)
609   $L57:
610       .set     noreorder
```

```
611       nop
612       .set    reorder
613   $L62:
614       lw  $v0,__sF+4
615       addu    $v0,$v0,-1
616       sw  $v0,__sF+4
617       bgez    $v0,$L65
618       la  $a0,__sF
619       la  $t9,__srget
620       jal $ra,$t9
621       sw  $v0,1076($fp)
622       b   $L66
623   $L65:
624       la  $v0,__sF
625       lw  $v1,0($v0)
626       move    $a0,$v1
627       lbu $a0,0($a0)
628       sw  $a0,1076($fp)
629       addu    $v1,$v1,1
630       sw  $v1,0($v0)
631   $L66:
632       lw  $v1,1076($fp)
633       sw  $v1,28($fp)
634       li  $v0,-1              # 0xffffffffffffffff
635       bne $v1,$v0,$L64
636       b   $L63
637   $L64:
638       lw  $v0,28($fp)
639       sll $v1,$v0,2
640       addu    $v0,$fp,24
641       addu    $v0,$v1,$v0
642       addu    $v0,$v0,16
643       lw  $v0,0($v0)
644       beq $v0,$zero,$L68
645       lw  $v1,32($fp)
646       lw  $v0,28($fp)
647       bne $v1,$v0,$L68
648       b   $L62
649   $L68:
650       lw  $a0,28($fp)
651       la  $a1,__sF+88
652       la  $t9,__sputc
653       jal $ra,$t9
654       move    $v1,$v0
655       li  $v0,-1              # 0xffffffffffffffff
656       bne $v1,$v0,$L69
657       la  $t9,errorPorEscritura
658       jal $ra,$t9
659       sw  $v0,1068($fp)
660       b   $L52
661   $L69:
662       lw  $v0,28($fp)
663       sw  $v0,32($fp)
664       b   $L62
665   $L63:
666       sw  $zero,1068($fp)
667   $L52:
668       lw  $v0,1068($fp)
669       move    $sp,$fp
670       lw  $ra,1088($sp)
671       lw  $fp,1084($sp)
672       addu    $sp,$sp,1096
673       j   $ra
674       .end    sString1
675       .size   sString1, .-sString1
676       .align  2
677       .globl  string1String2
678       .ent    string1String2
679   string1String2:
680       .frame  $fp,328,$ra     # vars= 288, regs= 3/0, args= 16, extra= 8
681       .mask   0xd0000000,-8
682       .fmask  0x00000000,0
683       .set    noreorder
684       .cpload $t9
```

```
685       .set    reorder
686       subu    $sp,$sp,328
687       .cprestore 16
688       sw  $ra,320($sp)
689       sw  $fp,316($sp)
690       sw  $gp,312($sp)
691       move    $fp,$sp
692       sw  $a0,328($fp)
693       sw  $a1,332($fp)
694       addu    $v0,$fp,40
695       move    $a0,$v0
696       li  $a1,256         # 0x100
697       la  $t9,bzero
698       jal $ra,$t9
699       lw  $a0,328($fp)
700       la  $t9,strlen
701       jal $ra,$t9
702       sw  $v0,296($fp)
703       lw  $a0,332($fp)
704       la  $t9,strlen
705       jal $ra,$t9
706       sw  $v0,300($fp)
707       lw  $v1,296($fp)
708       lw  $v0,300($fp)
709       slt $v0,$v0,$v1
710       bne $v0,$zero,$L71
711       lw  $v0,296($fp)
712       sw  $v0,32($fp)
713       b   $L72
714   $L71:
715       lw  $v0,300($fp)
716       sw  $v0,32($fp)
717   $L72:
718       sw  $zero,28($fp)
719   $L73:
720       lw  $v0,28($fp)
721       lw  $v1,32($fp)
722       slt $v0,$v0,$v1
723       bne $v0,$zero,$L76
724       b   $L74
725   $L76:
726       lw  $v1,328($fp)
727       lw  $v0,28($fp)
728       addu    $v0,$v1,$v0
729       lb  $v1,0($v0)
730       addu    $v0,$fp,40
731       addu    $a0,$v0,$v1
732       lw  $v1,332($fp)
733       lw  $v0,28($fp)
734       addu    $v0,$v1,$v0
735       lbu $v0,0($v0)
736       sb  $v0,0($a0)
737       lw  $v0,28($fp)
738       addu    $v0,$v0,1
739       sw  $v0,28($fp)
740       b   $L73
741   $L74:
742       lw  $v1,32($fp)
743       lw  $v0,300($fp)
744       bne $v1,$v0,$L77
745       lw  $v0,300($fp)
746       sw  $v0,28($fp)
747   $L78:
748       lw  $v0,28($fp)
749       lw  $v1,296($fp)
750       slt $v0,$v0,$v1
751       bne $v0,$zero,$L81
752       b   $L77
753   $L81:
754       lw  $v1,328($fp)
755       lw  $v0,28($fp)
756       addu    $v0,$v1,$v0
757       lb  $v1,0($v0)
758       addu    $v0,$fp,40
```

```
759       addu    $a0,$v0,$v1
760       lw  $v1,332($fp)
761       lw  $v0,300($fp)
762       addu    $v0,$v1,$v0
763       addu    $v0,$v0,-1
764       lbu $v0,0($v0)
765       sb  $v0,0($a0)
766       lw  $v0,28($fp)
767       addu    $v0,$v0,1
768       sw  $v0,28($fp)
769       b   $L78
770   $L77:
771       .set    noreorder
772       nop
773       .set    reorder
774   $L82:
775       lw  $v0,__sF+4
776       addu    $v0,$v0,-1
777       sw  $v0,__sF+4
778       bgez    $v0,$L85
779       la  $a0,__sF
780       la  $t9,__srget
781       jal $ra,$t9
782       sw  $v0,308($fp)
783       b   $L86
784   $L85:
785       la  $v0,__sF
786       lw  $v1,0($v0)
787       move    $a0,$v1
788       lbu $a0,0($a0)
789       sw  $a0,308($fp)
790       addu    $v1,$v1,1
791       sw  $v1,0($v0)
792   $L86:
793       lw  $v1,308($fp)
794       sw  $v1,24($fp)
795       li  $v0,-1          # 0xffffffffffffffff
796       bne $v1,$v0,$L84
797       b   $L83
798   $L84:
799       addu    $v1,$fp,40
800       lw  $v0,24($fp)
801       addu    $v0,$v1,$v0
802       lb  $v0,0($v0)
803       bne $v0,$zero,$L87
804       lw  $a0,24($fp)
805       la  $a1,__sF+88
806       la  $t9,__sputc
807       jal $ra,$t9
808       move    $v1,$v0
809       li  $v0,-1          # 0xffffffffffffffff
810       bne $v1,$v0,$L82
811       la  $t9,errorPorEscritura
812       jal $ra,$t9
813       sw  $v0,304($fp)
814       b   $L70
815   $L87:
816       addu    $v1,$fp,40
817       lw  $v0,24($fp)
818       addu    $v0,$v1,$v0
819       lb  $v0,0($v0)
820       move    $a0,$v0
821       la  $a1,__sF+88
822       la  $t9,__sputc
823       jal $ra,$t9
824       move    $v1,$v0
825       li  $v0,-1          # 0xffffffffffffffff
826       bne $v1,$v0,$L82
827       la  $t9,errorPorEscritura
828       jal $ra,$t9
829       sw  $v0,304($fp)
830       b   $L70
831   $L83:
832       sw  $zero,304($fp)
```

```
833 $L70:
834     lw  $v0,304($fp)
835     move    $sp,$fp
836     lw  $ra,320($sp)
837     lw  $fp,316($sp)
838     addu    $sp,$sp,328
839     j   $ra
840     .end    string1String2
841     .size   string1String2, .-string1String2
842     .align  2
843     .globl  sString1String2
844     .ent    sString1String2
845 sString1String2:
846     .frame  $fp,328,$ra     # vars= 288, regs= 3/0, args= 16, extra= 8
847     .mask   0xd0000000,-8
848     .fmask  0x00000000,0
849     .set    noreorder
850     .cpload $t9
851     .set    reorder
852     subu    $sp,$sp,328
853     .cprestore 16
854     sw  $ra,320($sp)
855     sw  $fp,316($sp)
856     sw  $gp,312($sp)
857     move    $fp,$sp
858     sw  $a0,328($fp)
859     sw  $a1,332($fp)
860     sw  $zero,24($fp)
861     sw  $zero,28($fp)
862     li  $v0,-1          # 0xffffffffffffffff
863     sw  $v0,32($fp)
864     sw  $zero,36($fp)
865     addu    $v0,$fp,40
866     move    $a0,$v0
867     li  $a1,256         # 0x100
868     la  $t9,bzero
869     jal $ra,$t9
870     lw  $a0,328($fp)
871     la  $t9,strlen
872     jal $ra,$t9
873     sw  $v0,296($fp)
874     lw  $a0,332($fp)
875     la  $t9,strlen
876     jal $ra,$t9
877     sw  $v0,300($fp)
878     lw  $v1,296($fp)
879     lw  $v0,300($fp)
880     slt $v0,$v0,$v1
881     bne $v0,$zero,$L92
882     lw  $v0,296($fp)
883     sw  $v0,36($fp)
884     b   $L93
885 $L92:
886     lw  $v0,300($fp)
887     sw  $v0,36($fp)
888 $L93:
889     sw  $zero,28($fp)
890 $L94:
891     lw  $v0,28($fp)
892     lw  $v1,36($fp)
893     slt $v0,$v0,$v1
894     bne $v0,$zero,$L97
895     b   $L95
896 $L97:
897     lw  $v1,328($fp)
898     lw  $v0,28($fp)
899     addu    $v0,$v1,$v0
900     lb  $v1,0($v0)
901     addu    $v0,$fp,40
902     addu    $a0,$v0,$v1
903     lw  $v1,332($fp)
904     lw  $v0,28($fp)
905     addu    $v0,$v1,$v0
906     lbu $v0,0($v0)
```

```
907         sb   $v0,0($a0)
908         lw   $v0,28($fp)
909         addu     $v0,$v0,1
910         sw   $v0,28($fp)
911         b    $L94
912     $L95:
913         lw   $v1,36($fp)
914         lw   $v0,300($fp)
915         bne  $v1,$v0,$L98
916         lw   $v0,300($fp)
917         sw   $v0,28($fp)
918     $L99:
919         lw   $v0,28($fp)
920         lw   $v1,296($fp)
921         slt  $v0,$v0,$v1
922         bne  $v0,$zero,$L102
923         b    $L98
924     $L102:
925         lw   $v1,328($fp)
926         lw   $v0,28($fp)
927         addu     $v0,$v1,$v0
928         lb   $v1,0($v0)
929         addu     $v0,$fp,40
930         addu     $a0,$v0,$v1
931         lw   $v1,332($fp)
932         lw   $v0,300($fp)
933         addu     $v0,$v1,$v0
934         addu     $v0,$v0,-1
935         lbu  $v0,0($v0)
936         sb   $v0,0($a0)
937         lw   $v0,28($fp)
938         addu     $v0,$v0,1
939         sw   $v0,28($fp)
940         b    $L99
941     $L98:
942         .set     noreorder
943         nop
944         .set     reorder
945     $L103:
946         lw   $v0,__sF+4
947         addu     $v0,$v0,-1
948         sw   $v0,__sF+4
949         bgez     $v0,$L106
950         la   $a0,__sF
951         la   $t9,__srget
952         jal  $ra,$t9
953         sw   $v0,308($fp)
954         b    $L107
955     $L106:
956         la   $v0,__sF
957         lw   $v1,0($v0)
958         move     $a0,$v1
959         lbu  $a0,0($a0)
960         sw   $a0,308($fp)
961         addu     $v1,$v1,1
962         sw   $v1,0($v0)
963     $L107:
964         lw   $v1,308($fp)
965         sw   $v1,24($fp)
966         li   $v0,-1              # 0xffffffffffffffff
967         bne  $v1,$v0,$L105
968         b    $L104
969     $L105:
970         addu     $v1,$fp,40
971         lw   $v0,24($fp)
972         addu     $v0,$v1,$v0
973         lb   $v0,0($v0)
974         bne  $v0,$zero,$L108
975         addu     $v1,$fp,40
976         lw   $v0,32($fp)
977         addu     $v0,$v1,$v0
978         lb   $v1,0($v0)
979         lw   $v0,24($fp)
980         beq  $v1,$v0,$L103
```

```
981       lw   $a0,24($fp)
982       la   $a1,__sF+88
983       la   $t9,__sputc
984       jal  $ra,$t9
985       move     $v1,$v0
986       li   $v0,-1           # 0xffffffffffffffff
987       bne  $v1,$v0,$L110
988       la   $t9,errorPorEscritura
989       jal  $ra,$t9
990       sw   $v0,304($fp)
991       b    $L91
992   $L110:
993       lw   $v0,24($fp)
994       sw   $v0,32($fp)
995       b    $L103
996   $L108:
997       addu     $v1,$fp,40
998       lw   $v0,24($fp)
999       addu     $a0,$v1,$v0
1000      addu     $v1,$fp,40
1001      lw   $v0,32($fp)
1002      addu     $v0,$v1,$v0
1003      lb   $v1,0($a0)
1004      lb   $v0,0($v0)
1005      beq  $v1,$v0,$L103
1006      addu     $v1,$fp,40
1007      lw   $v0,24($fp)
1008      addu     $v0,$v1,$v0
1009      lb   $v0,0($v0)
1010      move     $a0,$v0
1011      la   $a1,__sF+88
1012      la   $t9,__sputc
1013      jal  $ra,$t9
1014      move     $v1,$v0
1015      li   $v0,-1           # 0xffffffffffffffff
1016      bne  $v1,$v0,$L113
1017      la   $t9,errorPorEscritura
1018      jal  $ra,$t9
1019      sw   $v0,304($fp)
1020      b    $L91
1021  $L113:
1022      lw   $v0,24($fp)
1023      sw   $v0,32($fp)
1024      b    $L103
1025  $L104:
1026      sw   $zero,304($fp)
1027  $L91:
1028      lw   $v0,304($fp)
1029      move     $sp,$fp
1030      lw   $ra,320($sp)
1031      lw   $fp,316($sp)
1032      addu     $sp,$sp,328
1033      j    $ra
1034      .end     sString1String2
1035      .size    sString1String2, .-sString1String2
1036      .align   2
1037      .globl   dsString1String2
1038      .ent     dsString1String2
1039  dsString1String2:
1040      .frame   $fp,2120,$ra         # vars= 2080, regs= 3/0, args= 16, extra
              = 8
1041      .mask    0xd0000000,-8
1042      .fmask   0x00000000,0
1043      .set     noreorder
1044      .cpload  $t9
1045      .set     reorder
1046      subu     $sp,$sp,2120
1047      .cprestore 16
1048      sw   $ra,2112($sp)
1049      sw   $fp,2108($sp)
1050      sw   $gp,2104($sp)
1051      move     $fp,$sp
1052      sw   $a0,2120($fp)
1053      sw   $a1,2124($fp)
```

```
1054        sw   $zero,32($fp)
1055        addu     $v0,$fp,40
1056        move     $a0,$v0
1057        la   $t9,inicializarArrayBoolean
1058        jal $ra,$t9
1059        addu     $v0,$fp,1064
1060        move     $a0,$v0
1061        la   $t9,inicializarArrayBoolean
1062        jal $ra,$t9
1063        lw   $a0,2120($fp)
1064        la   $t9,strlen
1065        jal $ra,$t9
1066        sw   $v0,2088($fp)
1067        lw   $a0,2124($fp)
1068        la   $t9,strlen
1069        jal $ra,$t9
1070        sw   $v0,2092($fp)
1071        sw   $zero,28($fp)
1072    $L115:
1073        lw   $v0,28($fp)
1074        lw   $v1,2088($fp)
1075        slt $v0,$v0,$v1
1076        bne $v0,$zero,$L118
1077        b    $L116
1078    $L118:
1079        lw   $v1,2120($fp)
1080        lw   $v0,28($fp)
1081        addu     $v0,$v1,$v0
1082        lb   $v0,0($v0)
1083        sll $v1,$v0,2
1084        addu     $v0,$fp,24
1085        addu     $v0,$v1,$v0
1086        addu     $v1,$v0,16
1087        li   $v0,1               # 0x1
1088        sw   $v0,0($v1)
1089        lw   $v0,28($fp)
1090        addu     $v0,$v0,1
1091        sw   $v0,28($fp)
1092        b    $L115
1093    $L116:
1094        sw   $zero,28($fp)
1095    $L119:
1096        lw   $v0,28($fp)
1097        lw   $v1,2092($fp)
1098        slt $v0,$v0,$v1
1099        bne $v0,$zero,$L122
1100        b    $L120
1101    $L122:
1102        lw   $v1,2124($fp)
1103        lw   $v0,28($fp)
1104        addu     $v0,$v1,$v0
1105        lb   $v0,0($v0)
1106        sll $v1,$v0,2
1107        addu     $v0,$fp,24
1108        addu     $v0,$v1,$v0
1109        addu     $v1,$v0,1040
1110        li   $v0,1               # 0x1
1111        sw   $v0,0($v1)
1112        lw   $v0,28($fp)
1113        addu     $v0,$v0,1
1114        sw   $v0,28($fp)
1115        b    $L119
1116    $L120:
1117        .set     noreorder
1118        nop
1119        .set     reorder
1120    $L123:
1121        lw   $v0,__sF+4
1122        addu     $v0,$v0,-1
1123        sw   $v0,__sF+4
1124        bgez     $v0,$L126
1125        la   $a0,__sF
1126        la   $t9,__srget
1127        jal $ra,$t9
```

```
1128        sw   $v0,2100($fp)
1129        b    $L127
1130    $L126:
1131        la   $v0,__sF
1132        lw   $v1,0($v0)
1133        move     $a0,$v1
1134        lbu  $a0,0($a0)
1135        sw   $a0,2100($fp)
1136        addu     $v1,$v1,1
1137        sw   $v1,0($v0)
1138    $L127:
1139        lw   $v1,2100($fp)
1140        sw   $v1,24($fp)
1141        li   $v0,-1              # 0xffffffffffffffff
1142        bne  $v1,$v0,$L125
1143        b    $L124
1144    $L125:
1145        lw   $v0,24($fp)
1146        sll  $v1,$v0,2
1147        addu     $v0,$fp,24
1148        addu     $v0,$v1,$v0
1149        addu     $v0,$v0,16
1150        lw   $v0,0($v0)
1151        bne  $v0,$zero,$L123
1152        lw   $v0,24($fp)
1153        sll  $v1,$v0,2
1154        addu     $v0,$fp,24
1155        addu     $v0,$v1,$v0
1156        addu     $v0,$v0,1040
1157        lw   $v0,0($v0)
1158        beq  $v0,$zero,$L130
1159        lw   $v1,32($fp)
1160        lw   $v0,24($fp)
1161        bne  $v1,$v0,$L130
1162        b    $L123
1163    $L130:
1164        lw   $a0,24($fp)
1165        la   $a1,__sF+88
1166        la   $t9,__sputc
1167        jal  $ra,$t9
1168        move     $v1,$v0
1169        li   $v0,-1              # 0xffffffffffffffff
1170        bne  $v1,$v0,$L131
1171        la   $t9,errorPorEscritura
1172        jal  $ra,$t9
1173        sw   $v0,2096($fp)
1174        b    $L114
1175    $L131:
1176        lw   $v0,24($fp)
1177        sw   $v0,32($fp)
1178        b    $L123
1179    $L124:
1180        sw   $zero,2096($fp)
1181    $L114:
1182        lw   $v0,2096($fp)
1183        move     $sp,$fp
1184        lw   $ra,2112($sp)
1185        lw   $fp,2108($sp)
1186        addu     $sp,$sp,2120
1187        j    $ra
1188        .end     dsString1String2
1189        .size    dsString1String2, .-dsString1String2
1190        .align   2
1191        .globl   inicializarArrayBoolean
1192        .ent     inicializarArrayBoolean
1193    inicializarArrayBoolean:
1194        .frame   $fp,24,$ra      # vars= 8, regs= 2/0, args= 0, extra= 8
1195        .mask    0x50000000,-4
1196        .fmask   0x00000000,0
1197        .set     noreorder
1198        .cpload  $t9
1199        .set     reorder
1200        subu     $sp,$sp,24
1201        .cprestore 0
```

```
1202        sw   $fp,20($sp)
1203        sw   $gp,16($sp)
1204        move     $fp,$sp
1205        sw   $a0,24($fp)
1206        sw   $zero,8($fp)
1207    $L133:
1208        lw   $v0,8($fp)
1209        slt  $v0,$v0,256
1210        bne  $v0,$zero,$L136
1211        b    $L132
1212    $L136:
1213        lw   $v0,8($fp)
1214        sll  $v1,$v0,2
1215        lw   $v0,24($fp)
1216        addu     $v0,$v1,$v0
1217        sw   $zero,0($v0)
1218        lw   $v0,8($fp)
1219        addu     $v0,$v0,1
1220        sw   $v0,8($fp)
1221        b    $L133
1222    $L132:
1223        move     $sp,$fp
1224        lw   $fp,20($sp)
1225        addu     $sp,$sp,24
1226        j    $ra
1227        .end     inicializarArrayBoolean
1228        .size    inicializarArrayBoolean, .-inicializarArrayBoolean
1229        .rdata
1230        .align   2
1231    $LC12:
1232        .ascii   "Usage:\n\000"
1233        .align   2
1234    $LC13:
1235        .ascii   "tp0 -h\n\000"
1236        .align   2
1237    $LC14:
1238        .ascii   "tp0 -V\n\000"
1239        .align   2
1240    $LC15:
1241        .ascii   "tp0 [options] string1 string2\n\000"
1242        .align   2
1243    $LC16:
1244        .ascii   "tp0 [options] string1\n\000"
1245        .align   2
1246    $LC17:
1247        .ascii   "Options:\n\000"
1248        .align   2
1249    $LC18:
1250        .ascii   "-V, --version\n\000"
1251        .align   2
1252    $LC19:
1253        .ascii   "-h, --help\n\000"
1254        .align   2
1255    $LC20:
1256        .ascii   "-d, --delete\n\000"
1257        .align   2
1258    $LC21:
1259        .ascii   "-s, --squeeze\n\000"
1260        .text
1261        .align   2
1262        .globl   mostrarH
1263        .ent     mostrarH
1264    mostrarH:
1265        .frame   $fp,40,$ra      # vars= 0, regs= 3/0, args= 16, extra= 8
1266        .mask    0xd0000000,-8
1267        .fmask   0x00000000,0
1268        .set     noreorder
1269        .cpload  $t9
1270        .set     reorder
1271        subu     $sp,$sp,40
1272        .cprestore 16
1273        sw   $ra,32($sp)
1274        sw   $fp,28($sp)
1275        sw   $gp,24($sp)
```

```
1276        move    $fp,$sp
1277        la  $a0,$LC12
1278        la  $t9,printf
1279        jal $ra,$t9
1280        la  $a0,$LC13
1281        la  $t9,printf
1282        jal $ra,$t9
1283        la  $a0,$LC14
1284        la  $t9,printf
1285        jal $ra,$t9
1286        la  $a0,$LC15
1287        la  $t9,printf
1288        jal $ra,$t9
1289        la  $a0,$LC16
1290        la  $t9,printf
1291        jal $ra,$t9
1292        la  $a0,$LC17
1293        la  $t9,printf
1294        jal $ra,$t9
1295        la  $a0,$LC18
1296        la  $t9,printf
1297        jal $ra,$t9
1298        la  $a0,$LC19
1299        la  $t9,printf
1300        jal $ra,$t9
1301        la  $a0,$LC20
1302        la  $t9,printf
1303        jal $ra,$t9
1304        la  $a0,$LC21
1305        la  $t9,printf
1306        jal $ra,$t9
1307        move    $sp,$fp
1308        lw  $ra,32($sp)
1309        lw  $fp,28($sp)
1310        addu    $sp,$sp,40
1311        j   $ra
1312        .end    mostrarH
1313        .size   mostrarH, .-mostrarH
1314        .rdata
1315        .align  2
1316    $LC22:
1317        .ascii  "ERROR:␣ocurre␣un␣error␣de␣escritura\000"
1318        .text
1319        .align  2
1320        .globl  errorPorEscritura
1321        .ent    errorPorEscritura
1322    errorPorEscritura:
1323        .frame  $fp,40,$ra      # vars= 0, regs= 3/0, args= 16, extra= 8
1324        .mask   0xd0000000,-8
1325        .fmask  0x00000000,0
1326        .set    noreorder
1327        .cpload $t9
1328        .set    reorder
1329        subu    $sp,$sp,40
1330        .cprestore 16
1331        sw  $ra,32($sp)
1332        sw  $fp,28($sp)
1333        sw  $gp,24($sp)
1334        move    $fp,$sp
1335        la  $a0,$LC22
1336        la  $t9,perror
1337        jal $ra,$t9
1338        li  $v0,-1          # 0xffffffffffffffff
1339        move    $sp,$fp
1340        lw  $ra,32($sp)
1341        lw  $fp,28($sp)
1342        addu    $sp,$sp,40
1343        j   $ra
1344        .end    errorPorEscritura
1345        .size   errorPorEscritura, .-errorPorEscritura
1346        .align  2
1347        .ent    __sputc
1348    __sputc:
1349        .frame  $fp,48,$ra      # vars= 8, regs= 3/0, args= 16, extra= 8
```

```
1350        .mask   0xd0000000,-8
1351        .fmask  0x00000000,0
1352        .set    noreorder
1353        .cpload $t9
1354        .set    reorder
1355        subu    $sp,$sp,48
1356        .cprestore 16
1357        sw  $ra,40($sp)
1358        sw  $fp,36($sp)
1359        sw  $gp,32($sp)
1360        move    $fp,$sp
1361        sw  $a0,48($fp)
1362        sw  $a1,52($fp)
1363        lw  $v1,52($fp)
1364        lw  $v0,52($fp)
1365        lw  $v0,8($v0)
1366        addu    $v0,$v0,-1
1367        sw  $v0,8($v1)
1368        bgez    $v0,$L3
1369        lw  $v0,52($fp)
1370        lw  $v1,52($fp)
1371        lw  $a0,8($v0)
1372        lw  $v0,24($v1)
1373        slt $v0,$a0,$v0
1374        bne $v0,$zero,$L2
1375        lb  $v1,48($fp)
1376        li  $v0,10          # 0xa
1377        bne $v1,$v0,$L3
1378        b   $L2
1379    $L3:
1380        lw  $a1,52($fp)
1381        lw  $v1,0($a1)
1382        lbu $a0,48($fp)
1383        move    $v0,$v1
1384        sb  $a0,0($v0)
1385        andi    $v0,$a0,0x00ff
1386        addu    $v1,$v1,1
1387        sw  $v1,0($a1)
1388        sw  $v0,24($fp)
1389        b   $L1
1390    $L2:
1391        lw  $a0,48($fp)
1392        lw  $a1,52($fp)
1393        la  $t9,__swbuf
1394        jal $ra,$t9
1395        sw  $v0,24($fp)
1396    $L1:
1397        lw  $v0,24($fp)
1398        move    $sp,$fp
1399        lw  $ra,40($sp)
1400        lw  $fp,36($sp)
1401        addu    $sp,$sp,48
1402        j   $ra
1403        .end    __sputc
1404        .size   __sputc, .-__sputc
1405        .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

# 8. Conclusiones

El presente logró ser una introducción a distintas herramientas que se utilizarán en la materia. Entre las actividades realizadas podemos resaltar:

- Compilar código C utilizando gcc tanto en Linux como en NetBSD.
- Obtener el assembly generado por el compilador para la arquitectura MIPS.
- Utilizar el emulador GXemul para emular una maquina MIPS.
- Utilizar un tunel SSH tanto para trabajar en la consola del guest como para transferir archivos del host al guest y viceversa.
- Desarrollar el presente informe en LATEX.